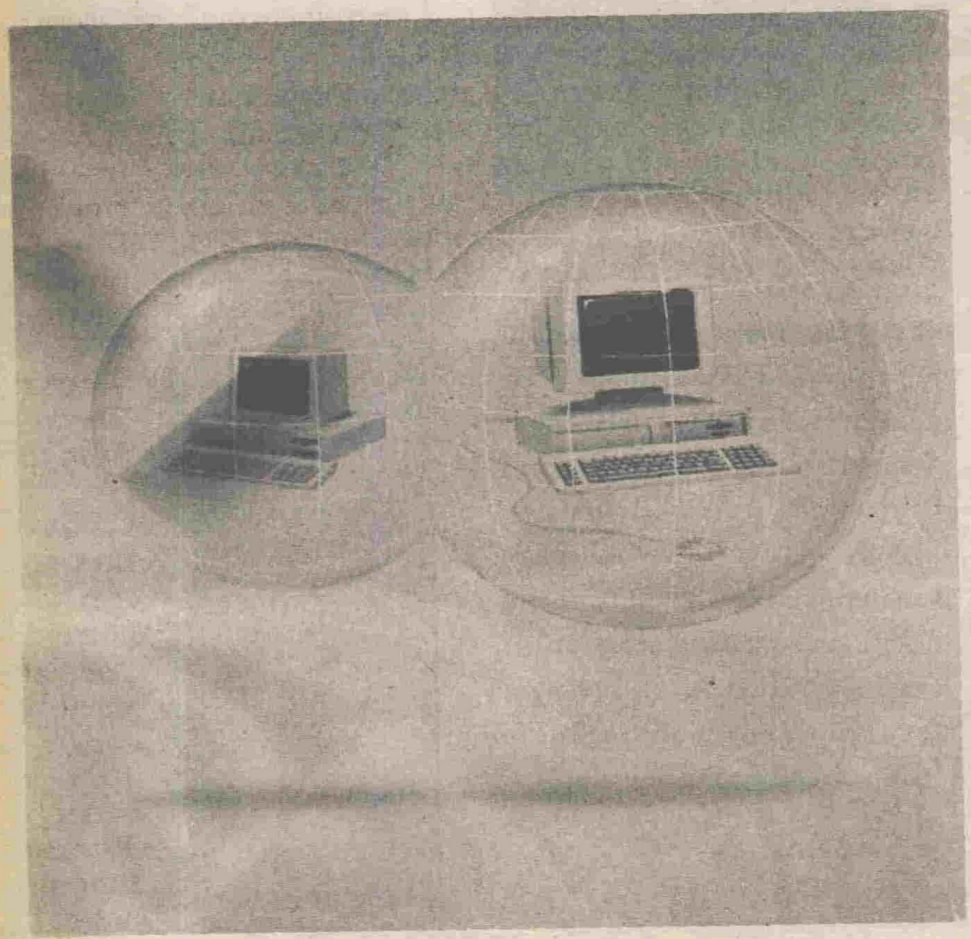
# คุยเฟื่องเรื่อง TCP/IP

*สิ่งละอันพันละน้อย เกี่ยวกับสถาปัตยกรรมการส่งข้อมูล TCP/IP มาเล่าสู่กันฟัง*



## บทนำ

เทคโนโลยีทางด้านคอมพิวเตอร์รุดหน้าไปอย่างรวดเร็วตลอดเวลา ในสมัยแรก ๆ เราเล่นไมโครคอมพิวเตอร์ก็ว่ากันเครื่องเดียวโดด ๆ เล่นดีเบส โลตัสไปได้เรื่อย แต่ในปัจจุบันทุกคนเริ่มเห็นความสำคัญของการแบ่งกันใช้ข้อมูลข่าวสารร่วมกัน มีการทำงานเป็นทีมในแบบเวอร์กกรุ๊ป (Work group) มีระบบ MIS สำหรับผู้บริหาร สิ่งเหล่านี้เพิ่มความสำคัญให้กับการสื่อสารข้อมูลด้วยคอมพิวเตอร์มากขึ้นทุกที เดี๋ยวนี้คนเริ่มรู้จักเน็ตแวร์ 3 Com, แอบเปิ้ลทอล์คกันมากขึ้น ผลิตภัณฑ์เหล่านั้นเป็นของในโลกของไมโครคอมพิวเตอร์ แต่ในโลกของมินิคอมพิวเตอร์และเครื่องเวอร์กสเตชันที่เริ่มจะมีบทบาทต่อเรามากขึ้นนั้น การส่งผ่านข้อมูลจะเป็นไปตามมาตรฐานของ TCP/IP เกือบทั้งหมด

ในบทความนี้ ผมจะเอาเรื่องสิ่งละอันพันละน้อยเกี่ยวกับ TCP/IP มาเล่าสู่กันฟังอย่างเรื่อยเปื่อยตามใจคนเขียน อ่านบทความนี้จบลงแล้วจะได้เอาไปคุยกับคนอื่นได้อย่างสนุกสนานยิ่งขึ้น

## แนวคิดเกี่ยวกับการเชื่อมต่อระหว่างข่ายงานคอมพิวเตอร์

ในสมัยนี้ความก้าวหน้าในเรื่องของคอมพิวเตอร์เน็ตเวอร์กเป็นไปอย่างมาก ทุกคนคงจะคุ้นเคยกับ LAN และอาจจะได้ยินมาตรฐานอื่น เช่น X.25

ที่ใช้ใน Wide Area Network มาบ้างแล้ว มาตรฐานเหล่านี้คืออะไร?

โดยปกติคอมพิวเตอร์ที่คุยกันผ่านข่ายสื่อสารนั้น จะคุยกันรู้เรื่องได้ต้องเป็นไปตามข้อตกลงอันหนึ่งซึ่งรู้กัน เหมือนกับเราคุยกับฝรั่งก็ต้องใช้ภาษาอังกฤษ ข้อตกลงนี้ต้องกำหนดรายละเอียดของฮาร์ดแวร์ต่าง ๆ โปรโตคอล (Protocol) ซึ่งหมายถึง รูปแบบของข้อมูลที่จะส่ง รวมไปถึงขั้นตอนต่าง ๆ ในการส่งผ่านข้อมูล ข้อตกลงเหล่านี้เองคือมาตรฐานในการสื่อสารข้อมูลที่ทำให้ผู้ผลิตอุปกรณ์ต่าง ๆ เช่น LAN, WAN ผลิตอุปกรณ์ที่มาคุยกันได้รู้เรื่อง แต่สิ่งที่กล่าวมาแล้วเป็นการมองในแง่ของฮาร์ดแวร์เท่านั้น

ในแง่ของซอฟต์แวร์ทางด้านการสื่อสารข้อมูล การพัฒนาซอฟต์แวร์เป็นเรื่องที่ยุ่งยากมาก ถ้าเรามีคอมพิวเตอร์อยู่แค่สองสามตัวแล้วอยู่บนข่ายสื่อสารอันเดียวกันก็คงจะไม่เป็นอะไรเท่าไหร่ ในซอฟต์แวร์ก็ใส่ไปว่าคุยกับเครื่องไหนต่ออยู่ที่ไหนเลย ในสมัยแรก ๆ เขาก็ทำกันแบบนี้ ปรากฏว่าไม่ได้เรื่องเพราะเราจะเผชิญปัญหาหลายประการ คือ

1. ในข่ายงานคอมพิวเตอร์ขนาดใหญ่มีการเปลี่ยนแปลงอยู่ตลอดเวลา การเขียนซอฟต์แวร์ต้องมีการปรับกันเรื่อย นอกจากนั้นการเพิ่มความสามารถในระบบหมายถึง การแก้ไขซอฟต์แวร์กันขนานใหญ่

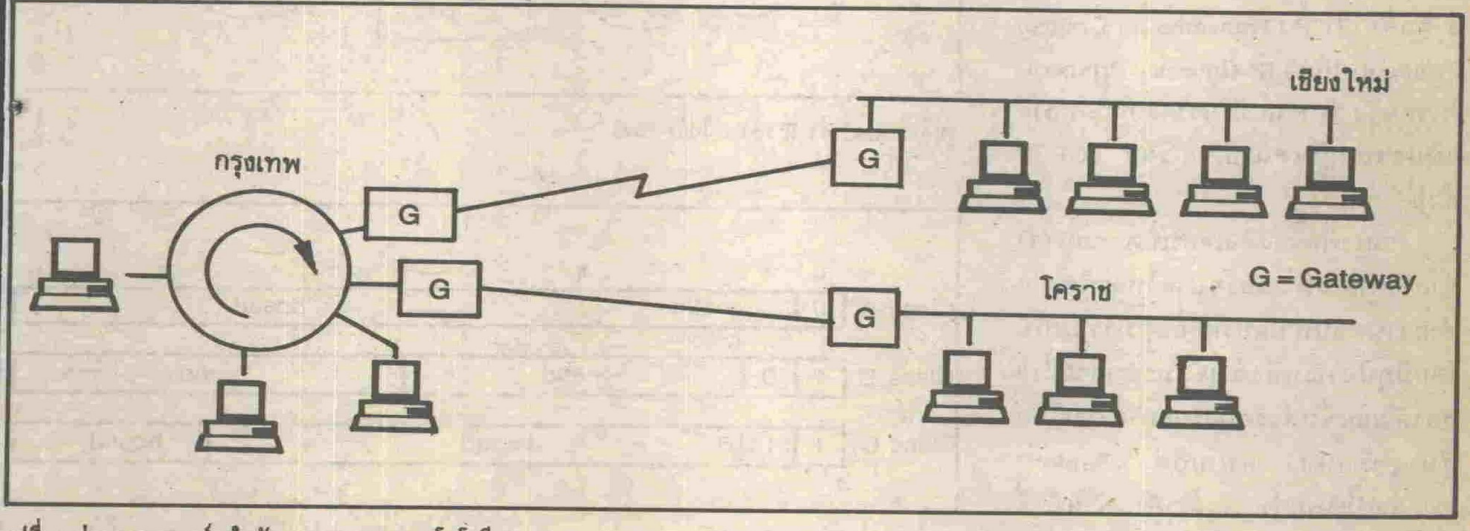
2. ข่ายงานคอมพิวเตอร์ข่ายเดียวไม่สามารถรองรับความต้องการของผู้ใช้ได้ ยกตัวอย่างเช่น บริษัทขายประกันแห่งหนึ่งมีแลนใช้ภายในสำนักงานที่กรุงเทพ วันดีคืนดีมีสำนักงานสาขาที่เชียงใหม่กับโคราช จะเห็นว่าแลนอย่างเดียววิ่งไม่ถึงแน่ ต้องเชื่อมแลน 2 ข่ายเข้าด้วยกันด้วยโมเด็มหรือใช้ THAIPAK ซึ่งเป็นข่าย X.25 ของการสื่อสาร แถมแลนแต่ละสาขาอาจเป็นคนละแบบ ที่กรุงเทพอาจเป็นโทเก้นริง 16 Mbit เชียงใหม่เป็นอีเธอร์เนต ถ้าซอฟต์แวร์ต้องมารองรับรายละเอียดพวกนี้ตายแน่ครับ ดังรูปที่ 1

3. ผู้ใช้ทุกคนฝันจะคุยกับใครก็ได้ในโลก เหมือนกับไปรษณีย์ เราเขียนชื่อที่อยู่ของผู้รับปลายทางซึ่งเขาอยู่ที่ไหนก็ได้ในโลกแล้วไปทิ้งลงตู้ไปรษณีย์ จดหมายควรจะถึงปลายทางเอง เราไม่เคยต้องรู้ว่าการสื่อสารส่งจดหมายไปยังไง ลงรถ ขึ้นเรือ ขึ้นเครื่องบิน ไปเมืองไหนบ้าง ผู้ใช้ข่ายงานคอมพิวเตอร์ฝันถึงอิเล็กทรอนิกส์เมล์ที่ส่งถึงใครก็ได้ในโลก นั่งอยู่ที่บ้านไปแวบ login ใช้เครื่องที่อเมริกาได้ฟรี ๆ

เราลองมาฝันกันดูอย่างที่อินเตอร์เนตเอ็นจิเนียร์ที่ออกแบบ TCP/IP เขาฝันกันมาก่อนว่า สถาปัตยกรรมของข่ายงานคอมพิวเตอร์นั้นควรจะทำอะไรให้เราบ้าง

## ข่ายงานคอมพิวเตอร์ในฝัน (ของผม)

1. คนเขียนซอฟต์แวร์ไม่ต้องรู้รูปร่าง (Topoloty) จำนวน รวมถึงวิธีการที่ข่ายงานทั้งหลายต่อกันอยู่ เราอ้างถึงคอมพิวเตอร์เครื่องใดก็ได้ในโลกด้วยตำแหน่ง (Address) ก็สามารถส่งข้อมูลไปยังคอมพิวเตอร์เครื่องนั้นได้ ระบบปฏิบัติการ (Operating System) และข่ายงานจะเป็นคนแก้ปัญหานี้ให้เอง ไม่ว่าจะเป็นเรื่องของการหาเส้นทาง (Routing)




รูปที่ 1 ข่ายงานขององค์กรในลักษณะผสมหลายเทคโนโลยี

การตรวจสอบความผิดพลาด (Error detection and recovery) และอื่น ๆ

2. เราไม่จำกัดยี่ห้อของคอมพิวเตอร์ เทคโนโลยีของข่ายงานคอมพิวเตอร์ย่อย ๆ ที่เอามาต่อกัน จำนวนคอมพิวเตอร์ในข่าย เรามี IBM, DEC, SUN, HP สารพัน ต่อกันผ่าน X.25, FDDI, โทเก้นริง อีเธอร์เนต ปนกันได้ ผู้ใช้จะเป็นเพียงข่ายงานนามธรรม (ABSTRACT Network) เพียงข่ายเดียวที่เชื่อมต่อคอมพิวเตอร์ทั้งหมดเข้าด้วยกัน

เนื่องจากเราเป็นคนดีเลยได้รับรางวัลให้เกิดฝันที่เป็นจริงโดยมีคนคิดทำข่ายงานในฝันไว้ให้เราเสร็จแล้ว หน้าตาเป็นอย่างไรผมจะอธิบายให้ฟังต่อไป

## แนวคิดของอินเตอร์เนต

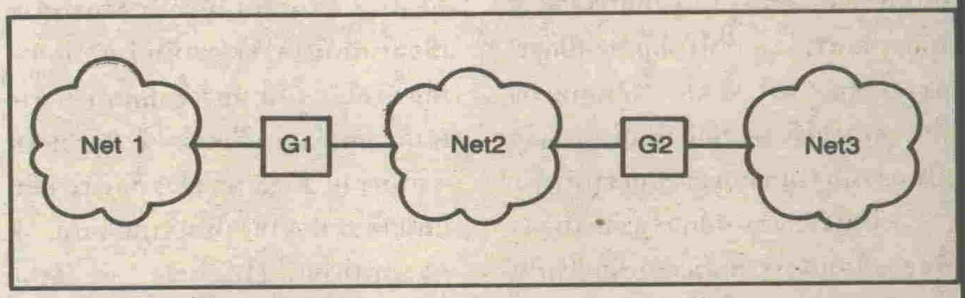
การเชื่อมต่อข่ายงานคอมพิวเตอร์จำนวนมากเข้าด้วยกันเรียกว่า Internetworking หรืออินเตอร์เนต เป็นคำที่เราจะพบบ่อยและใช้ปนกับ TCP/IP ผมเลยขออธิบายตรงนี้นิดว่าอินเตอร์เนตเป็นการเชื่อมข่ายซึ่งทำได้หลายวิธี TCP/IP เป็นสถาปัตยกรรมข่ายงานอันหนึ่งซึ่งดังมาจากโปรโตคอลสำคัญ 2 ตัวคือ TCP (Transmission Control Protocol) และ IP (Internet Protocol) ในชุดของ TCP/IP มีโปรโตคอลอีกหลายแบบรวมอยู่ในสถาปัตยกรรม TCP/IP ด้วย

แนวคิดของอินเตอร์เนตจะมองว่าข่ายงานคอมพิวเตอร์ขนาดใหญ่เกิดจากข่ายงานคอมพิวเตอร์ย่อย ๆ มารวมกันโดยมีอุปกรณ์หนึ่งซึ่งเชื่อมระหว่างข่ายสองตัวและรับส่งข้อมูลในลักษณะบล็อกสั้น ๆ เรียกว่า แพคเก็ต (Packet) อุปกรณ์นี้เรียกว่า เกตเวย์ ทำหน้าที่เชื่อมต่อข่ายงานและแปลงข้อมูลระหว่างสองข่ายงานซึ่งอาจใช้เทคโนโลยีต่างกัน เช่น อีเธอร์เนตหรือโทเก้นริง เป็นต้น ดังรูปที่ 2

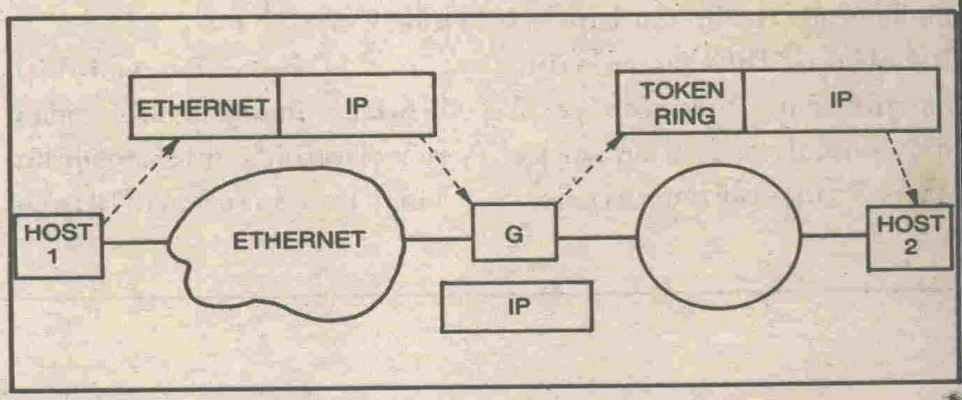
ในแนวคิดของอินเตอร์เนตกำหนดให้ข้อมูลส่งในรูปแบบของ Internet Packet (IP) ซึ่งไม่ขึ้นกับฟิสิคัลเน็ตเวอร์กย่อย ๆ (อยู่ในส่วนที่เป็นข้อมูลของฟิสิคัลเน็ตเวอร์กแพคเก็ต เกตเวย์จะเป็นตัวดึง IP มาพิจารณาแล้วส่งต่อกันไปจนถึงปลายทางเอง ดังรูปที่ 3

การทำงานในลักษณะนี้ผู้ใช้จะมองเห็นข่ายงานคอมพิวเตอร์เป็น "เสมือน" ข่ายใหญ่เพียงข่ายเดียว ซึ่งอ้างตำแหน่งด้วยอินเตอร์เนตแอดเดรสขนาด 32 บิต

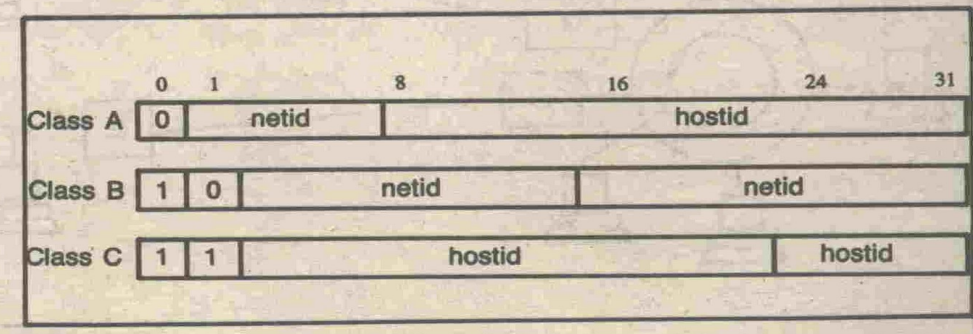
อินเตอร์เนตแอดเดรสนี้มีหน้าตาดังรูปที่ 4 แบ่งออกเป็น 3 ส่วน ส่วนแรกจะบอกคลาสของข่ายงานคอมพิวเตอร์ว่าเป็น A,B,C ถ้า A ก็ใหญ่หน่อย




รูปที่ 2 ข่ายงานย่อยที่เชื่อมกันผ่านเกตเวย์




รูปที่ 3 Host1 ส่ง IP Packet ไปยัง Host2




รูปที่ 4 รูปแบบของอินเตอร์เนตแอดเดรส

ถ้า C ก็เล็กนิดเดียว (มีได้แค่ 255 Host ในข่ายนั้น) อีกสองส่วนจะบอกว่าเป็นข่ายงานหมายเลขอะไรและคอมพิวเตอร์เครื่องไหนในข่ายงานนั้น

ปกติเราเขียนอินเตอร์เนตแอดเดรส หรือ IP แอดเดรสเป็นแบบที่เรียกว่า Dolted Decimal Notation เช่น 128.10.2.30 แต่ละเลขจะแทน 8 บิต แต่เขียนเป็นฐานสิบแทน

ผู้เขียนได้ลงตัวอย่างของการต่อเน็ตเวอร์กมาให้ดูตัวอย่างหนึ่งเป็นของมหาวิทยาลัย Purdue ที่อเมริกา ข่ายงานคอมพิวเตอร์ของภาควิชาคอมพิวเตอร์ของเขาจะเป็นดังรูปที่ 5 คอมพิวเตอร์แต่ละตัวจะมีชื่อตามตัวละครในนิยายอัศวินโต๊ะกลม ถ้าเป็นไทยคงมาจากรามเกียรติ์ อิเหนาละครับ

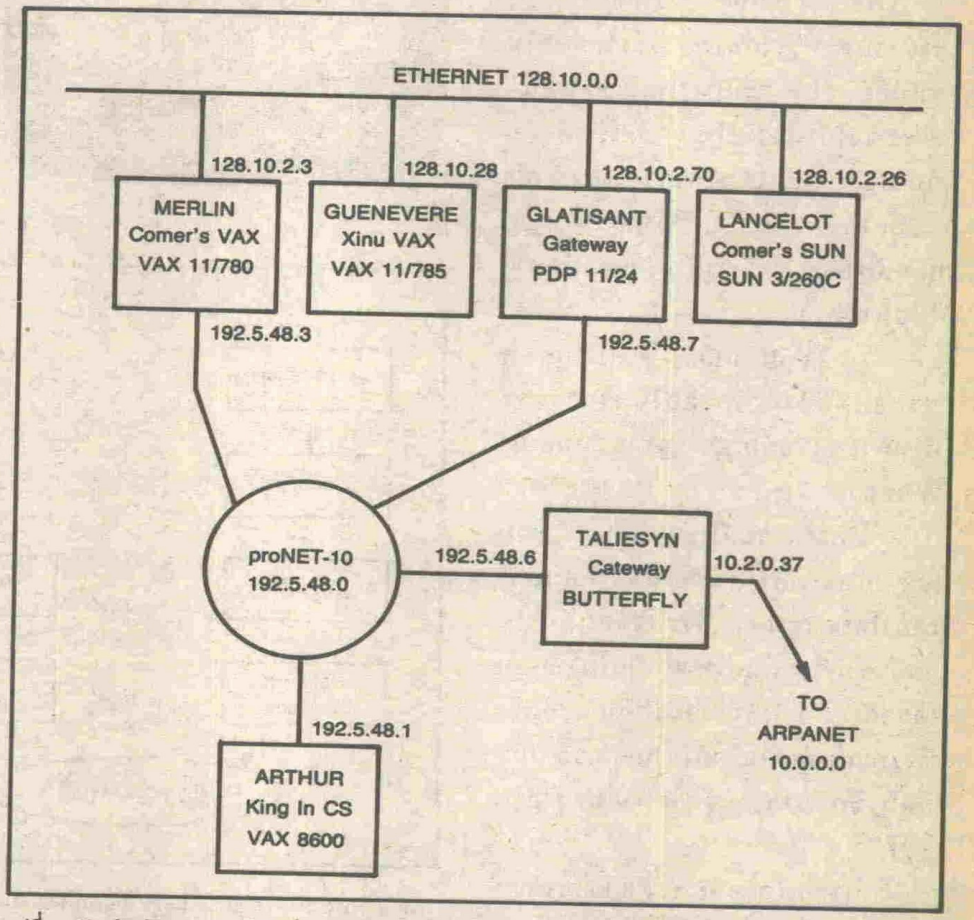
## สถาปัตยกรรมของ TCP/IP

จากแนวคิดพื้นฐานที่เราพูดกันมาทั้งหมด สถาปัตยกรรมของ TCP/IP จะเป็นเช่นไร คำตอบก็ออกมาในรูปที่ 6 แสดงสถาปัตยกรรมของ TCP/IP ซึ่งแบ่งออกเป็นชั้น ๆ ทั้งหมด 5 ชั้น แต่ละชั้นทำหน้าที่ต่าง ๆ กัน ดังนี้

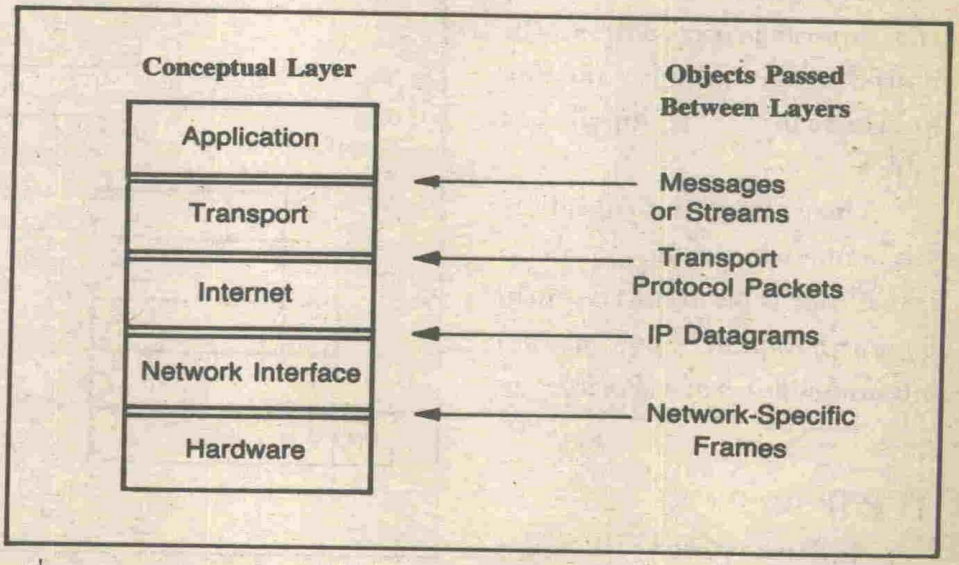
1. Hardware และ Network Interface ส่วนนี้คือข่ายงานย่อยที่อยู่เบื้องล่างเช่น Ethernet, X.25 ซึ่งทำหน้าที่ส่งเฟรมเฉพาะข่ายระหว่างคอมพิวเตอร์ในข่ายเดียวกันและเกตเวย์ด้วย

2. Internet Layer คือ IP packet ที่เราพูดถึงนั่นเอง ในชั้นนี้จะให้บริการที่เรียกว่า Connectionless Packet Delivery Service คือส่งข้อมูล 1 แพคเก็ต จากต้นทางไปปลายทาง ในชั้นนี้การส่งข้อมูลจะไม่ตรวจเรื่องการสูญหายหรือผิดพลาด คือไม่ Reliable แต่จะทำให้เกิดภาพของข่ายงานเสมือนขนาดใหญ่เท่านั้น

3. Transport Layer ในชั้นนี้จะให้บริการในการส่งข้อมูล ในลักษณะ




รูปที่ 5 ตัวอย่างข่ายงานคอมพิวเตอร์ภาควิชาวิทยาการคอมพิวเตอร์มหาวิทยาลัย Purdue สหรัฐอเมริกา




รูปที่ 6 สถาปัตยกรรมของ TCP/IP

Reliable Stream Transport Service ซึ่งจะรับผิดชอบในการจัดลำดับข้อมูล ตรวจสอบความผิดพลาดในการส่งผ่านข่ายงานระหว่างกลาง โปรโตคอลในชั้นนี้คือ TCP จะเป็น Transport Protocal ซึ่งโปรแกรมประยุกต์ใช้ติดต่อกันกับ UDP (User Datagram Packet) ซึ่งส่งข้อมูลในลักษณะเป็นดาต้าแกรม หรือแพคเก็ตเดี่ยว ๆ วิ่งจากโปรแกรมต้นทางไปปลายทาง

4. Application คือโปรแกรมประยุกต์ของเราที่เรียกใช้บริการทางด้านการส่งรับข้อมูลผ่านข่ายงานคอมพิวเตอร์

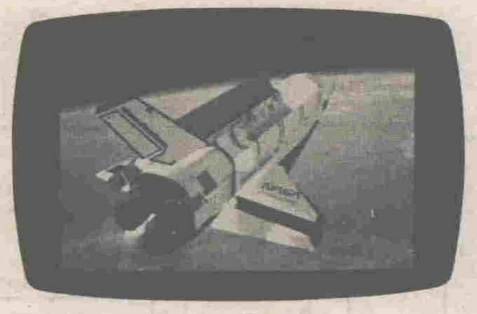
ในแต่ละระดับชั้นของ TCP/IP จะคุยกับระดับชั้นเดียวกันของคอมพิวเตอร์เครื่องอื่นผ่านการส่งข้อมูล ในระดับชั้นเดียวกันจะส่งข้อมูลในรูปแบบเดียวกัน ยกเว้นเน็ตเวอร์กอินเตอร์เฟส ซึ่งรูปแบบต่างกันได้ถ้าผ่านเกตเวย์ แนวคิดนี้จะอธิบายเป็นรูปได้ดังแสดงไว้ในรูปที่ 7

บางคนอาจจะสงสัยว่าไอ้สถาปัตยกรรมที่ส่งข้อมูลเป็นชั้น ๆ เวลาส่งจริงเรียงข้อมูลอย่างไร คำตอบก็คือโปรโตคอลของชั้นสูงขึ้นไปจะอยู่ในส่วนที่เป็นข้อมูลชั้นต่ำกว่า ดังที่เราเคยคุยไว้ในเรื่องการส่ง IP แพคเก็ต ดังรูปที่ 8

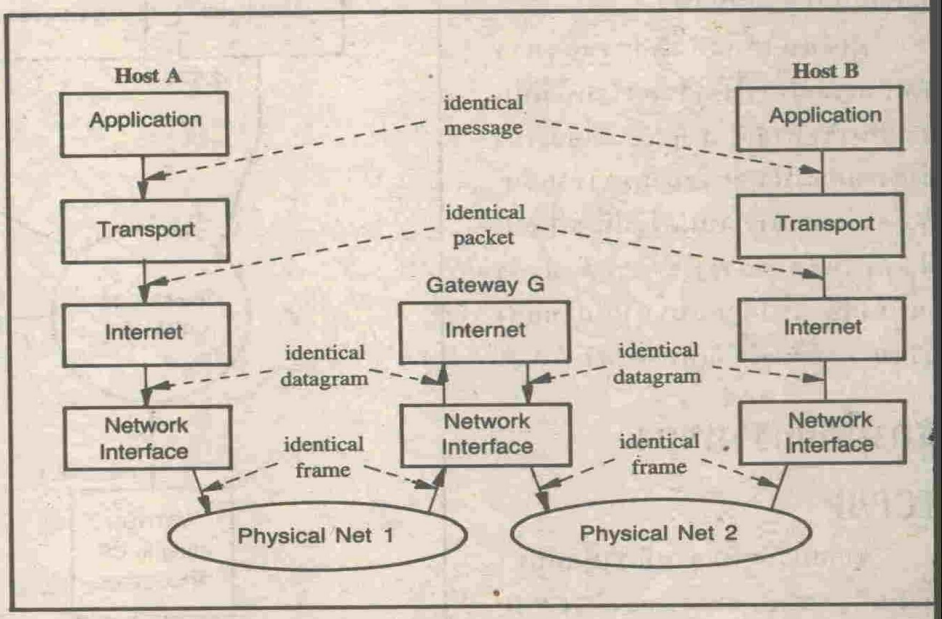
เวลาส่งลงจากชั้นสูงไปต่ำก็แปะโปรโตคอลชั้น ๆ มาเรื่อย ๆ เวลาส่งถึงเครื่องปลายทางวิ่งจากชั้นล่างขึ้นบนไปยังแอพพลิเคชันค่อย ๆ ถอดทีละชิ้น ๆ ไปจนเหลือแต่ยูสเซอร์ดาต้าตัวเปล่า ๆ

## TCP/IP และ OSI

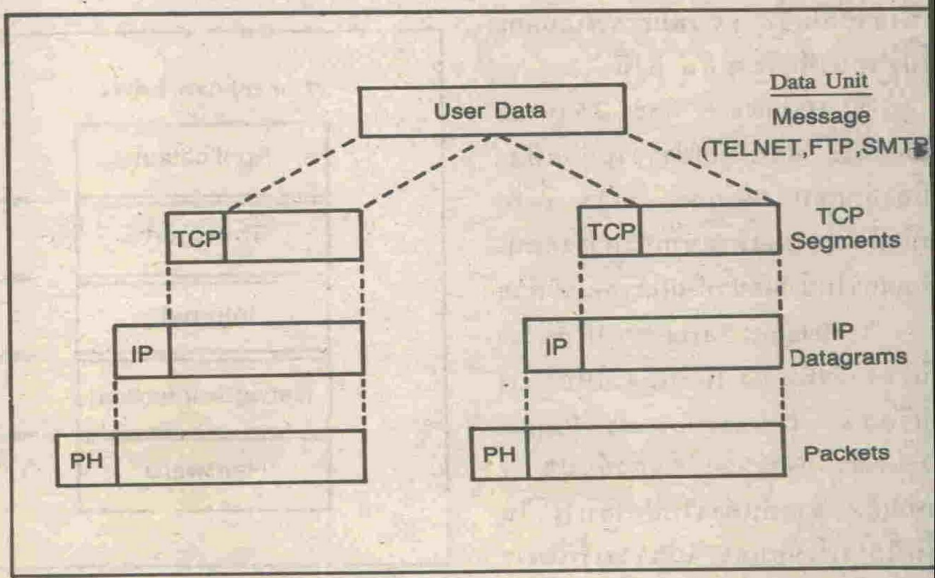
ในด้านสถาปัตยกรรมของข่ายคอมพิวเตอร์แล้ว คนที่สนใจองจะรู้จัก



OSI (Open System Interconnectio
ของ ISO เป็นอย่างดี OSI เป็นมาตรฐา
สากลที่ได้รับความสนใจและถูกผลักดั
ให้ใช้กันทั่วโลก โดยวางแนวคิดสำหรั
ข่ายงานคอมพิวเตอร์ที่เป็น "อิสระ
จากผู้ขาย แล้วยังรวมเทคโนโลยีต่าง
ไว้มากมาย แบ่งออกเป็น 7 ชั้น จะ




รูปที่ 7 การส่งผ่านข้อมูลระหว่างชั้นในสถาปัตยกรรมของ TCP/IP




รูปที่ 8 โปรโตคอลของชั้นบน ๆ จะแฝงไว้ในส่วนข้อมูลของชั้นล่าง

มาอธิบายในโอกาสต่อไป) ถ้าเทียบ 2 ความคิดจะเป็นไปในรูปแบบดังรูปที่ 9

ตามความเห็นส่วนตัวของผม TCP/IP นั้นผ่านการทดสอบมานานและใช้จริงได้ดี ขณะที่ OSI ยังเป็นของใหม่ แต่ในแนวคิดแบบนามธรรม OSI จะลึกซึ้งกว่ามาก (SAP, CEP, Entites, PDU, SDU)

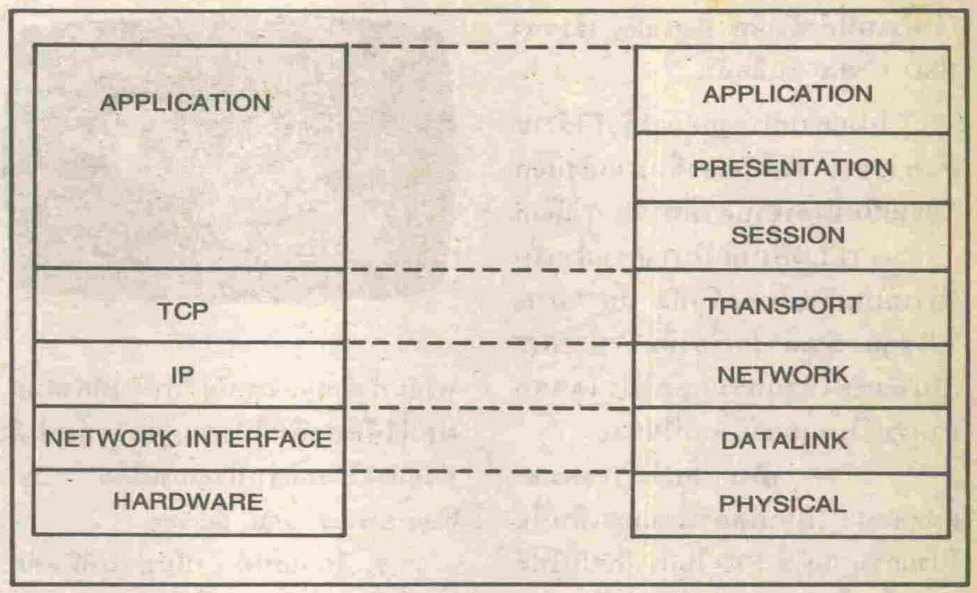
จุดที่ต่างกันก็คือ OSI ต้องการ Reliable Service ในระดับเน็ตเวอร์กแล้วเป็นแบบ Connection Oriented เพราะ OSI มาจากแรงผลักทางยุโรป ซึ่งชินกับ Wide Area Network เช่น X.25 แต่ TCP/IP ต้องการแค่ Unreliable deliverly ในระดับ IP การทำ Error Recovery ผลักเป็นภาระในระดับ TRANSPORT บน Host แทน

แนวคิดของ OSI เน็ตเวอร์ก จะเก่ง แต่ Host ไม่เก่งมากก็ได้ แต่ TCP/IP ผลักภาระการทำ Reliable Service ออกจากเน็ตเวอร์กไปสู่ Host ในปัจจุบันความคิดนี้ OSI ได้รับมาบางส่วนมาใช้กับแลนที่ทำงานแบบ OSI

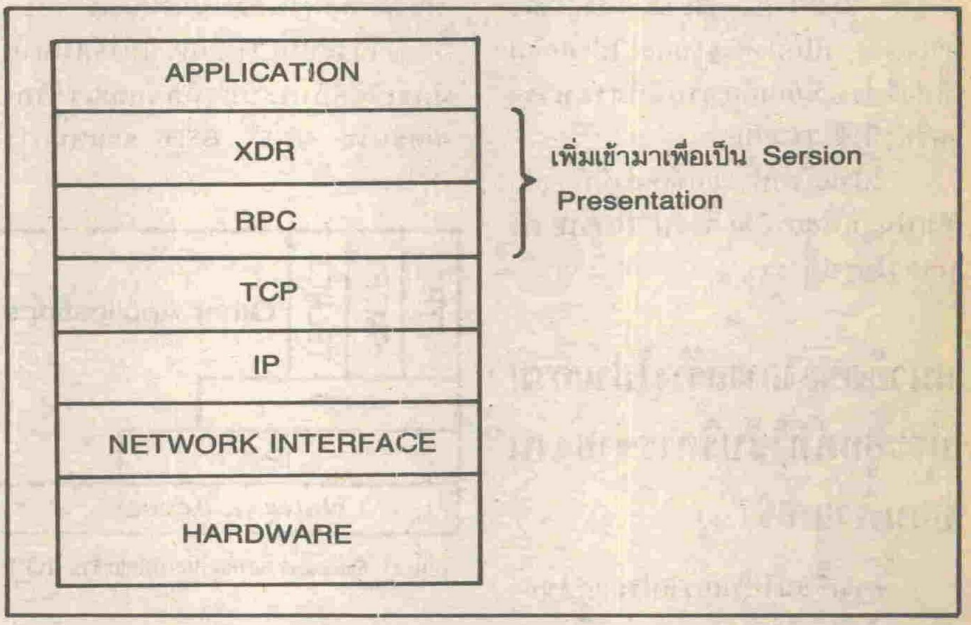
ข้อแตกต่างอีกประการคือ หน้าที่ของ Session, Presentation layer ไม่ได้ถูกกำหนดไว้แอพพลิเคชันต้องทำเองหมด

บางหน่วยงานจึงคิดโครงสร้างที่ไม่เป็นทางการ เพื่อเพิ่มชั้นที่ทำงานคล้ายกับ Session และ Presentation เข้าไปใน TCP/IP เช่น SUN Micro System จะใช้โครงสร้างลักษณะนี้ ดังรูปที่ 10

RPC (Remote Procedure Call) จะถูกนำมาแทน Session layer เพื่อประสานงานระหว่างโปรแกรม 2 ตัวในระดับสูง




รูปที่ 9 เปรียบเทียบ TCP/IP และ OSI




รูปที่ 10 แสดงการเพิ่มเติมส่วนที่จัดการกับ Session และ Presentation ใน TCP/IP

XDR (Xternal Data Representation) เป็นตัวแปลงรูปแบบข้อมูลเพื่อให้คอมพิวเตอร์ที่เก็บข้อมูลต่างกันติดต่อสื่อสารกันได้

## โปรแกรมประยุกต์ที่สำคัญใน TCP/IP

พัฒนาการของ TCP/IP ส่วนหนึ่งเกิดจากนักวิจัยในมหาวิทยาลัยโดย

เจ้าตำรับที่สำคัญคือ Berkeley เจ้าของ BSD UNIX อันลือลั่น

โปรแกรมประยุกต์บางตัวที่ใช้งานกันกว้างขวางได้พัฒนาขึ้นมาแล้วแจกกันจนเป็นมาตรฐานตัวที่สำคัญ ๆ ได้แก่

- TELNET เป็นโปรแกรมสำหรับทำงานเป็นรีโมทเทอร์มินัล คือ Login ไปใช้คอมพิวเตอร์เครื่องอื่น TELNET เป็นทั้งชื่อโปรแกรมและโปรโตคอลสำหรับรีโมทเทอร์มินัลที่ใช้ด้วย

- FTP เป็น File Transfer Protocal ในอินเตอร์เนตและเป็นชื่อโปรแกรม เราใช้ FTP ในการก๊อปปี้ไฟล์จากเครื่องอื่นในข่ายงานที่เจ้าของอนุญาตหรือเผลอตัว

- SMTP (Simple Mail Transfer Protocal) เป็นมาตรฐานและโปรแกรมที่ส่งไปรษณีย์อิเล็กทรอนิกส์ระหว่างเครื่องที่ใช้ TCP/IP

โปรแกรมประยุกต์เหล่านี้มีความสัมพันธ์กับสถาปัตยกรรม TCP/IP ดังแสดงในรูปที่ 11

## แนวคิดของการสร้างโปรแกรมประยุกต์ที่ใช้บริการข่ายงานคอมพิวเตอร์

การเขียนโปรแกรมประยุกต์ของเราเองไม่ได้เป็นเรื่องยากอย่างที่หลายคนเข้าใจ ในการออกแบบโปรแกรมในสภาวะที่เป็นแบบกระจายเช่นนี้ มักจะใช้แนวคิดของ Client-Server เป็นหลัก

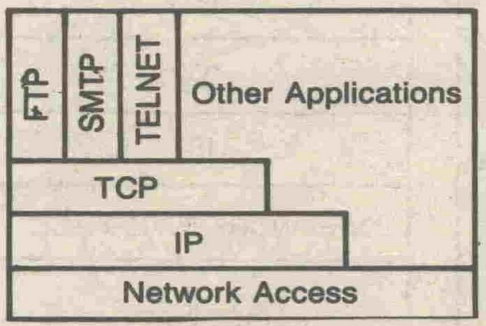
แนวคิดในการออกแบบในลักษณะของ Client-Server นี้แบ่งโปรแกรมออกเป็น 2 พวกคือ

1. เซอร์ฟเวอร์ พวกนี้จะเป็นคนรอรับการขอ (Request) จากไคลเอนต์

พร้อมทั้งข้อมูล จากนั้นก็ทำงานที่กำหนดผลที่ได้ก็ส่งคืนไปทางฝากไคลเอนต์ ตัวอย่างโปรแกรมประเภทนี้คือ Database Server, SQL Server

2. ไคลเอนต์ เป็นส่วนที่ติดต่อกับผู้ใช้ รับคำสั่งจากผู้ใช้แล้วแปลงเป็นข้อมูลส่งไปขอรับบริการจากเซอร์ฟเวอร์ เมื่อได้ผลก็หวนกลับมาฝากผู้ใช้

ในระบบ TCP/IP นี้ที่ใช้กันมากมักจะใช้คู่กับระบบยูนิกซ์และเวอร์กสเตชันใน UNIX BSD จะแทนการ




รูปที่ 11 Application ที่สำคัญในสถาปัตยกรรม TCP/IP

เชื่อมต่อในระหว่างโปรแกรมด้วยสิ่งที่เรียกว่า Socket ดังรูปที่ 12

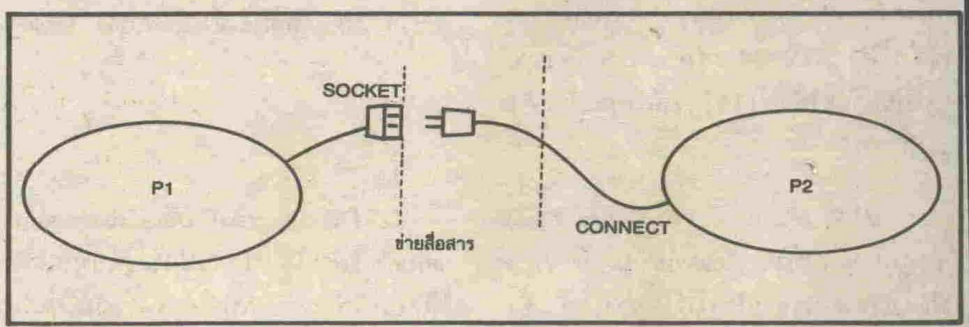
โปรแกรมเซอร์ฟเวอร์จะสร้างซอคเก็ตรอไว้โดยใช้ UNIX System Call โปรแกรมไคลเอนต์จะของคอนเน็ตซึ่งเปรียบเหมือนการเสียบปลั๊กให้เกิดการเชื่อมต่อระหว่างโปรแกรม การเชื่อมต่อที่เกิดขึ้นจะใช้ส่งข้อมูลไปมาได้ รายละเอียดมากกว่านี้คงต้องค้นกันเอง สำหรับผู้สนใจ ผมจะให้ตัวอย่างเซอร์ฟเวอร์และไคลเอนต์ที่ผมเขียนไว้ในตัวอย่างท้ายบทความก็แล้วกันครับ

โปรแกรมตัวอย่างนี้แบ่งเป็น 2 โปรแกรม Genderd ซึ่งเป็นเซอร์ฟเวอร์ เราจะสั่งให้ทำงานเป็นแบคกราวน์เพื่อบอกเพศของคอมพิวเตอร์ว่าเป็นชาย หญิง เกย์ ส่วน Gender จะเป็นคำสั่งที่ขอติดต่อกับเซอร์ฟเวอร์บนเครื่องใด ๆ เพื่อให้ผู้ใช้ถามเพศของคอมพิวเตอร์นั้นได้

## อ่านต่อได้ที่ไหนดี

จากแนวคิดคร่าว ๆ ที่เราพูดไปแล้วถ้าสนใจจะหาอ่านต่อได้จากที่ไหน ผมขอแนะนำหนังสือให้สัก 2 เล่มคือ

Internetworking with TCP/IP ของ Donglas Comer เป็นหนังสือที่ดีมาก อ่านเข้าใจง่ายแล้วสนุกด้วย ส่วน




รูปที่ 12 แนวความคิดเรื่องซอคเก็ต

อีกเล่มคือ Handbook of Computer Communications Standards Vol. 3 DOD Protocol standards. โดย Williom Stalling เรื่องของซอคเก็ตจะมีในหนังสือ Comer บ้างแล้วไปอ่านเพิ่มเอาจากคู่มือ ยูนิกซ์ที่มากับเครื่องคงจะได้การ หวังว่าผู้อ่านคงสนุกเหมือนผมนะครับ

รูปที่ 13 โปรแกรม GENDERDC

```
GENDERD.C                    Sunday, August 19, 1990


/*
 *   NAME
 *                      Gender daemon
 *   SYNOPSIS
 *            genderd  <MALE¦FEMALE>
 *   DESCRIPTION
 *       This program is a simple daemon that tell which sex this host is.
 *   It is useless server but demonstrate how to write TCP/IP server
 *   in networking environment.
 *   NOTE
 *       This server is designed and coded by Putchong Uthayopas.Any commen
t
 *   can directly forward to:
 *
 *      Putchong Uthayopas
 *      Department of computer engineering,Faculty of engineering
 *      Kasetsart university,Bangkok Thailand. 10700
 *      Tel. 5797566,5794584
 *      Any contribution will be very appreciated.
 *   DATE   13 August 1990
 *   SEE ALSO
 *        gender (also written by myself)
 */
#include  <stdio.h>
#include  <string.h>
#include  <sys/types.h>
#include  <sys/socket.h>
#include  <netinet/in.h>
#include  <netdb.h>

#define   GENDER_SERVICE        "gender"
#define   GENDER_PROTOCOL       "tcp"
#define   MAX_HOSTNAME          40
#define   QUEUE_LENGTH          5

static char mygender[20];
static char localhost[MAX_HOSTNAME];

int fatal_exit(),gender_service(),usage();

main(argc,argv)
int argc;
char *argv[];
{
   struct hostent *hp; /* Host entry in database */
   struct servent *sp; /* Service entry point */
   struct sockaddr_in sa,isa;
   int  isalen;
   int  isoc;

   /*   Check number of argument */
   if (argc != 2) usage(argv[0]);
   strcpy(mygender,argv[1]);
   /*
    *    Get gender service port
```

รูปที่ 13 (ต่อ)

```
GENDERD.C                    Sunday, August 19, 1990

     */
    if ( (sp= getservbyname(GENDER_SERVICE,GENDER_PROTOCOL)) == NULL) {
        fatal_exit(0);
    }
    /*
     *     get current host name
     */
    gethostname(localhost,MAX_HOSTNAME);
    if ( (hp= gethostbyname(localhost)) == NULL ) {
       fatal_exit(1);
    }
    /*
     *    Put the gender service adderss to socket
     */
     sa.sin_port = sp->s_sport;
     bcopy( (char *)hp->h_addr,(char *)&sa.sin_addr,hp->h_length);
     sa.sin_family=hp->h_addrtype;
    /*
     *    Create our socket
     */
     if ((s=socket(hp->h_addrtype,sock_stream,0)) < 0)' {
        perror("socket");
        fatal_exit(2);
     }
    /*
     *    Then bind service address to our socket
     */
     if (bind(s,&sa,sizeof(sa),0) < 0) {
         perror("bind");
         fatal_exit(3);
     }
     /*
      *    Listen to our socket
      */
     listen(s,QUEUE_LENGTH);
     /*
      *    Now wait and service request comming
      */
      for(;;) {
         isalen=sizeof(isa);- /* incoming address */
         if( (isoc=accept(s,&isa,&isalen)) < 0) {
            perror("Accept");
            fatal_exit(4);
         }
         gender_service(isoc);
         close(isoc);
      }

}
char buff[256];
gender_service(socket)
int socket;
{
     sprintf(buff,"HOST: %s  is  %s",localhost,mygender);
     write(socket,buff,strlen(buff));
```

รูปที่ 13 (ต่อ)

```
GENDERD.C                    Sunday, August 19, 1990


}

/*
 *                    usage
 *    if error then display usage of this software
 */
usage(name)
char *name;
{
     printf("usage: %s <male or female>\n",name);
     exit(1);
}
/*
 *    fatal_exit
 *    Exit and explain cause
 */
char *errmsg[]={"No service available"};
fatal_exit(code)
int code;
{
     if (code >=0 && code < sizeof(errmsg) ) {
        fprintf(stderr,"Fatal abort: %s\n",errmsg[code]);
     } else fprintf(stderr,"Invalid Fatal error code %d\n",code);
     exit(1);
}

GENDER.C                     Sunday, August 19, 1990                    Page 1



/*
 *    NAME
 *                        Gender client
 *    SYNOPSIS
 *              gender <Host name>
 *    DESCRIPTION
 *        This program is a simple client that tell which sex the host is.
 *    It perform its service by communicating with daemon process on that ho
st.
 *    NOTE
 *        This program is designed and coded by Putchong Uthayopas.Any comme
nt
 *    can directly forward to:
 *
 *       Putchong Uthayopas
 *       Department of computer engineering,Faculty of engineering
 *       Kasetsart university,Bangkok Thailand. 10700
 *       Tel. 5797566,5794584
 *       Any contribution will be very appreciated.
 *    DATE   13 August 1990
 *    SEE ALSO
 *         genderd
 */
#include  <stdio.h>
#include  <string.h>
#include  <sys/types.h>
#include  <sys/socket.h>
#include  <netinet/in.h>
#include  <netdb.h>                                                    →
```

รูปที่ 13 (ต่อ)

```
#define   GENDER_SERVICE        "gender"
#define   GENDER_PROTOCOL       "tcp"
#define   MAX_HOSTNAME          40
#define   QUEUE_LENGTH          5
#define   BUFF_SIZE             256


static char buff[BUFF_SIZE];
static char remotehost[MAX_HOSTNAME];

int fatal_exit(),gender_service(),usage();

main(argc,argv)
int argc;
char *argv[];
{
   struct hostent *hp; /* Host entry in database */
   struct servent *sp; /* Service entry point */
   struct sockaddr_in sa,isa;
   int  s;
   int  len;

   /*   Check number of argument */

GENDER.C                    Sunday, August 19, 1990


   if (argc != 2) usage(argv[0]);
   /* Get remote host name */
   strcpy(remotehost,argv[1]);
   /*
    *    Get gender service port
    */
   if ( (sp= getservbyname(GENDER_SERVICE,GENDER_PROTOCOL)) == NULL) {
       fatal_exit(0);
   }
   if ( (hp=gethostbyname(remotehost))== NULL){
        fatal_exit(1);
    }
   /*
    *   Put the gender service adderss to socket
    */
    sa.sin_port = sp->s_port;
    bcopy( (char *)hp->h_addr,(char *)&sa.sin_addr,hp->h_length);
    sa.sin_family=hp->h_addrtype;
   /*
    *   Create our socket
    */
    if ((s=socket(hp->h_addrtype,SOCK_STREAM,0)) < 0) {
       perror("socket");
       fatal_exit(2);
    }
    if (connect(s,&sa,sizeof(sa)) < 0) {
         perror("connect");
         fatal_exit(4);
    }
    while ( (len=read(s,buff,BUFF_SIZE)) > 0) {
        write(0,buff,len);
    }
    printf("\n");
    close(s);
    exit(0);
```

รูปที่ 13 (ต่อ)

```
}

/*
 *                    usage
 *   if error then display usage of this software
 */
usage(name)
char *name;
{
     printf("usage: %s <male or female>\n",name);
     exit(1);
}
/*
 *   fatal_exit
 *   Exit and explain cause
 */
char *errmsg[]={"No service available"};
fatal_exit(code)
int code;
{


GENDER.C                    Sunday, August 19, 1990


     if (code >=0 && code < sizeof(errmsg) ) {
        fprintf(stderr,"Fatal abort: %s\n",errmsg[code]);
     } else fprintf(stderr,"Invalid Fatal error code %d\n",code);
     exit(1);
}
```